REMARKS

Claims 1-49 are pending in the present application. By this Response, claims 1, 7-9, 12, 16, 27, 30-34, 37, 41, 42, 44-47 and 49 are amended for clarification. Independent claims 1, 16, 27, 41, 42 and 49 are amended to recite that the trace data is obtained from a trace of the execution of a computer program and that the trace data is provided in terms of machine code corresponding to the computer program. Independent claims 1, 27 and 42 are also amended to clarify that the symbolic data is data representative of human readable alphanumeric text associated with modules that are loaded in the computer system. In addition, other minor clarifying amendments are made to the claims to ensure consistency of terminology in the claims. Support for these clarifying amendments may be found at least at page 2, lines 26-28, page 4, lines 21-22, page 5, lines 3-7, page 23, lines 7-24, and page 24, line 32 to page 25, line 20. Reconsideration of the claims is respectfully requested.

Amendments are made to the specification to update cross-referenced patent application information on page 1 of the specification. No new matter has been added by any of the amendments to the specification.

I. Amendments to the Claims

Applicants respectfully submit that the amendments made to the claims by this Response are only clarifying in nature. The originally filed claims make reference to "trace data" and comparing trace data with module symbolic data in a merged symbol file. As set forth in the present specification, trace data is data that indicates execution flow for an executing program (page 2, lines 26-28). Furthermore, the present specification states that trace data is provided in terms of machine code, e.g., process identifiers, addresses, and the like (page 5, lines 4-7). It is further evident from other portions of the description of the present invention that the trace data is data that is obtained during the tracing of the execution of a computer program (e.g., see page 23, lines 7-24, and page 24, line 32 to page 25, line 20). Thus, it is Applicants' position that it is clear from the original claims that the use of the term "trace data" in the context of

the present specification refers to data that is obtained from a trace of an execution of a computer program and that the trace data is provided in terms of machine code. Therefore, the amendments to the claims do not add any new features to the claims.

Although Applicants believe that the originally filed claims, when read in light of the specification, include this understanding of the term "trace data," from the nature of the art cited by the Examiner, it is believe that the Examiner may have misinterpreted this term. Therefore, in an effort to expedite prosecution of the application and clarify the user of the term "trace data" in the claims, the independent claims have been amended to include the phrase "wherein the trace data is obtained from a trace of an execution of a computer program of which the module is a part, and wherein the trace data is provided in terms of machine code corresponding to the computer program.

Similarly, claims 1, 27 and 42 are amended to include the clarification that symbolic data is data representative of human readable alphanumeric text. Again, while Applicants believe that the originally filed claims, when read in light of the specification, include the understanding that symbolic data is data representative of human readable alphanumeric text, in an effort to expedite prosecution of this application, these claims are amended to include this definition.

II. 35 U.S.C. § 102, Alleged Anticipation

The Office Action rejects claims 1-49 under 35 U.S.C. § 102(b) as being allegedly anticipated by Baldwin, Jr. et al. (U.S. Patent No. 5,452,449). This rejection is respectfully traversed.

Claim 1, which is representative of the other rejected independent claims 16, 27, 41, 42 and 49 with regard to similarly recited subject matter, reads as follows:

> 1. A method of verifying symbolic data for modules that are loaded in a computer system, comprising:
>
> reading trace data for a module, wherein the trace data is obtained from a trace of an execution of a computer program of which the module is a part, and wherein the trace data is provided in terms of machine code corresponding to the computer program;

comparing the trace data with symbolic data stored in a merged symbol file, wherein the symbolic data is data representative of human readable alphanumeric text associated with modules that are loaded in the computer system; and

verifying the symbolic data in response to a determination that the trace data matches the symbolic data in the merged symbol file based on one or more predetermined criteria.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicants respectfully submit that Baldwin does not identically show every element of the claimed invention arranges as they are in the claims. Specifically, Baldwin does not teach reading trace data that is obtained from a trace of an execution of a computer program and which is provided in terms of machine code, comparing the trace data with symbolic data, or verifying that the trace data matches the symbolic data.

Baldwin is directed to a system and method for analyzing source code to generate databases of information regarding inter-module relationships of the source code. Specifically, the system and method of Baldwin uses a batch database load function that extracts information from each module's source code and remarks so as to make the information available to an interactive program analysis portion of the Baldwin system (column 3, lines 55-59; column 9, lines 47-51). The Baldwin system generates eight databases, as provided in Table 3 of Baldwin (columns 14-15). These eight databases include information about the remarks in the source code (Source Code Database), information about the data elements that are referenced with in the procedure-division source code (How-Referenced Database), information about the data elements that are used within the procedure-division source code to change the value of other variables in the module (How-Used Database), information about the data elements that have their

values set or changed within the procedure-division source code (How-Set Database), information about the copy library members used by each module (Copy-Member Database), information about the call-parameters for each called module (Call Parameter Database), information about the parameters used in the entry statements of modules called by other modules (Entry Parameter Database), and information about the parameters passed between called and calling modules to allow the interactive multi-module system to match up the call and entry-parameters and expand the parameters to display information about the sub-elements of the parameters (Parameter-Expansion Database).

These eight databases that Baldwin generates by extracting information from the source code during a batch database load operation are then used by an interactive program analysis system to provide different reports to a user based on user input. The interactive program analysis system is basically an interface for a user to obtain inter-module relationship reports generated from the eight databases. A user may decide what information is needed to resolve a particular problem, determine how many modules he/she wants to include in the scope of the analysis, create, store, edit and copy lists of modules that would likely be reused, etc. (see column 17, et. seq.).

Thus, Baldwin performs all of the analysis based on information extracted from source code. Baldwin has nothing to do with trace data obtained from a trace of the execution of a computer program that is provided in terms of machine code. Moreover, Baldwin has nothing to do with comparing trace data with symbolic data or verifying that trace data matches symbolic data, let alone symbolic data that is in a merged symbol file. To the contrary, all of the operations performed by Baldwin are performed on information extracted from source code, not data obtained from tracing the execution of a computer program. Therefore, Baldwin does not teach each and every feature independent claims **1**, 16, 27, 41, 42 and 49.

The Office Action alleges that Baldwin teaches reading trace data for a module in Figure 1, element 20, Figure 2, elements 30 and 32, Figure 3, element 31, and column 4, line 30 to column 5, line 63. Element 20 of Figure 1 is the overall interactive multi-module program analysis system of Baldwin. Element 30 in Figure 2 represents the selection of all modules which combine to form an application program (column 8, lines

21-29), i.e. the identification of the modules whose source code will be the basis for the information extracted by the batch database load function. Element 32 refers to the step of creating the eight databases from information extracted from the source code (column 8, lines 35-42). Element 31 in Figure 3 refers to the interactive program analysis system of Baldwin which is used to provide an interface for a user to obtain reports based on the information extracted from source code and stored in the eight databases described above. None of these elements have anything to do with trace data obtained from a trace of the execution of a computer program. Moreover, none of these elements are provided in terms of machine code.

Column 4, line 30 to column 5, line 63 merely describes the detailed element analysis, process analysis and impact analysis that may be performed using the interactive program analysis system of Baldwin. While this section of Baldwin details the various analytical techniques that may be applied to the information maintained in the eight databases, these techniques are still applied to information extracted from source code. Moreover, the data upon which these techniques operate is not provided in terms of machine code. Thus, there is no need to perform any symbolic resolution by comparing the data to symbolic data provided in a merged symbol file. Therefore, despite the Office Action's allegations to the contrary, Baldwin in fact does not teach, or even suggest, reading trace data that is obtained from a trace of the execution of a computer program and which is provided in terms of machine code, comparing the trace data with symbolic data in a merged symbol file, or verifying that the trace data matches the module symbolic data, as recited in independent claims 1, 16, 27, 41, 42 and 49.

In view of the above, Applicants respectfully submit that Baldwin does not teach each and every feature of independent claims 1, 16, 27, 41, 42 and 49 as is required under 35 U.S.C. § 102(b). At least by virtue of their dependency on claims 1, 16, 27, and 42, respectively, Baldwin does not teach each and every feature of dependent claims 2-15, 17-26, 28-40 and 43-48. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1-49 under 35 U.S.C. § 102(b).

In addition to being dependent upon their respective independent claims, dependent claims 2-15, 17-26, 28-40 and 43-48 recite additional features that are not taught by Baldwin. For example, with regard to claims 2, 18, 28 and 43, Baldwin does

not teach that one or more criteria for verifying that trace data matches symbolic data includes one or more of a checksum, a timestamp, a fully qualified path, and a segment size. The Office Action alleges that these features are taught in Figure 1, element 20, Figure 2, elements 30, 32, 34, 36, 38, 40, 42, 44, 46 and 48, Figure 3, element 31, and column 9, lines 19-23 of Baldwin. None of the elements in Figures 1-3 cited by the Office Action mention anything regarding any one of a checksum, a timestamp, a fully qualified path, or a segment size, let alone using such to verify that trace data matches symbolic data.

Column 9, lines 19-23 merely states that an update process begins by creating an audit trail database of module names, their version and the date their extract was added to the information databases and that the audit trail database is updated after every update of the database. This section of Baldwin is merely referring to data that is stored to identify the series of updates performed to the databases, i.e. an audit trail. There is nothing in this, or any other, section of Baldwin that teaches, or even suggests, to use one or more of a checksum, a timestamp, a fully qualified path, or a segment size to verify that trace data matches symbolic data. Thus, contrary to the allegations made by the Office Action, Baldwin, in actuality, does not teach the features of claims 2, 18, 28 and 43.

With regard to claims 3-4, 19-20 and 29, Baldwin does not teach that trace data is read from either of a trace buffer or a trace file. As discussed above, Baldwin has nothing to do with tracing of the execution of a computer program and thus, there is no need in Baldwin for a trace buffer or a trace file. Baldwin only operates on source code and is not concerned with tracing execution.

The Office Action alleges that the features of these claims are taught by Baldwin based on the same elements in Figures 1, 2 and 3 reference repeatedly above, and in column 1, lines 13-17 which reads as follows:

> The present invention relates to systems for analyzing computer high-level language program modules, including their inter-module relationships. It further relates to a system for creating databases used in such program analysis systems.
> (emphasis added)

Neither Figures 1-3 nor the above cited section state anything regarding a trace buffer or a trace file. To the contrary, Baldwin merely teaches analyzing source code to generate databases and then providing a user interface for obtaining information from the databases. Baldwin has nothing to do with traces of execution of a computer program, let alone reading trace data from a trace buffer or a trace file. The cited section of Baldwin only serves to further support Applicants' position in that the cited section states that Baldwin analyzes high-level language program modules. As is well known to those of ordinary skill in the art, a high-level language is a language such as COBOL, FORTRAN, C++, or other programming language and is not machine code which is used in executing computer programs. Thus, again, this section of Baldwin further emphasizes the fact that Baldwin only operates on source code in a high-level language and not on trace data of a trace of the execution of a computer program. Therefore, there teaching or even need in Baldwin of a trace buffer or a trace file.

Regarding claims 5, 6, 30 and 31, Baldwin does not teach that reading, comparing and verifying are performed dynamically as trace data is written to a trace buffer. As discussed at length above, Baldwin does not have anything to do with tracing the execution of a computer program and thus, does not operate on trace data. Therefore, Baldwin cannot operate dynamically as trace data is written to a trace buffer. Once again, the Office Action points to the same elements in Figures 1-3 and then cites column 1, lines 13-17, column 4, line 30 to column 5, line 63 and column 9, lines 19-23. Each of these sections of Baldwin have been discussed above and have been shown to not teach anything regarding trace data, a trace buffer, or performing the operations of reading trace data, comparing trace data with symbolic data, or verifying that the trace data matches the symbolic data.

With regard to claims 7, 8, 9, 32, 33, 34, 44, 45 and 46, Baldwin does not teach comparing a checksum and timestamp in trace data with a checksum and timestamp in symbolic data (claims 7, 32 and 44), comparing a fully qualified path in the trace data with a fully qualified path in symbolic data, if the checksum comparison results in no match (claims 8, 33 and 45), or comparing segment length in trace data with segment length in symbolic data, if the fully qualified path comparison results in no match (claims 9, 34 and 46). As with every other rejection, the Office Action points to the same

elements shown in Figures 1-3 which have nothing to do with checking to determine whether trace data matches symbolic data and thus, have nothing to do with the operations recited in these claims. In addition, the Office Action further points to the same sections of Baldwin discussed above, i.e. column 1, lines 13-17, column 4, line 30 to column 5, line 63 and column 9, lines 19-23. These sections of Baldwin have nothing to do with verifying that trace data matches symbolic data and definitely do not teach or even suggest the specific operations and series of operations recited in claims 7-9, 32-34 and 44-46.

Regarding claims 10-11, 21-22 and 35-36, Baldwin does not teach that the one or more criteria associated priority (claims 10, 21 and 35) or that the one more criteria include checksum and timestamp, a fully qualified path and a segment size, and wherein the checksum and timestamp have a highest priority and the segment size has a lowest priority (claims 11, 22 and 36). Yet again, the Office Action points to the same sections of Baldwin as used in every other rejection. Nowhere in any of these sections is there any mention of a priority at all, let alone a priority that is associated with one or more criteria that are used to verify that trace data matches symbolic data. Thus, once again, despite the allegations made by the Office Action, Baldwin in fact does not teach anything regarding the features recited in claims 10-11, 21-22 and 35-36.

With regard to claims 12-13, 23-24, 37-38 and 47-48, Baldwin does not teach that a merged symbol file includes a plurality of module entries and that comparing trace data with symbolic data includes identifying a module entry that is a best match with the trace data (claims 12, 23, 37 and 47) or that identifying a module entry that is a best match includes comparing trace data to each of the plurality of module entries and identifying one of the plurality of module entries as a best match based on which of the one or more criteria is used to verify the module entry (claims 13, 24, 38 and 48). Again, the Office Action points to the same elements in Figures 1-3 and the same sections of Baldwin cited in all of the other rejections to allegedly teach the specific features recited in these claims. Once again, Applicants respectfully submit that these sections of Baldwin have nothing to do with the features of the present claims. Furthermore, the Examiner has failed to show where any of the features are allegedly taught in these cited sections.

Basically, the rejection, as with the rejection of all the other dependent claims, amounts to a recitation of the claim and a parenthetical statement citing the same sections as every other rejection. This in no way satisfies the Examiner's burden of showing where every element of the claims is specifically taught in the reference. Thus, with these claims, and every other dependent claim rejected in a similar manner, the Examiner has failed to establish a case of anticipation based on the Baldwin reference.

The same is true of the rejection of dependent claims 14-15, 25-26 and 39-40. Again, the Examiner merely recites the features in the claim and then adds a parenthetical that is identical to every other parenthetical used in the rejections of every other dependent claim. Merely appearing to address the features of the claim is not sufficient. The Examiner must consider every element of the claim and specifically identify where that feature is taught in the reference in order to establish a case of anticipation. The Examiner has failed to do so in the rejections of the dependent claims and thus, has not satisfied his burden. Therefore, Applicants respectfully request withdrawal of the rejections of dependent claims 2-15, 17-26, 28-40 and 43-48.

III. Alleged Double Patenting

The Office Action rejects claims 1-49 under the judicially created doctrine of obviousness-type double patenting. As suggested by the Office Action, a Terminal Disclaimer in compliance with 37 CFR 1.321(c) is filed herewith disclaiming any extension of term beyond the term of U.S. Patent No. 6,658,416 which is owned by International Business Machines Corporation, assignee of the present application. Therefore, Applicants respectfully request withdrawal of the rejection of claims 1-49 based on alleged obviousness-type double patenting.

IV. Conclusion

It is respectfully urged that the subject application is patentable over Baldwin and Hussain and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: 11/1/04

Stephen J. Walder, Jr.
Reg. No. 41,534
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

Attachment:

Terminal Disclaimer